Semantic Interoperability: Case Study in Ontology-Based Solutions

Mike Pool (mpool@iet.com)

Information Extraction and Transport, Inc.

August 25, 2004

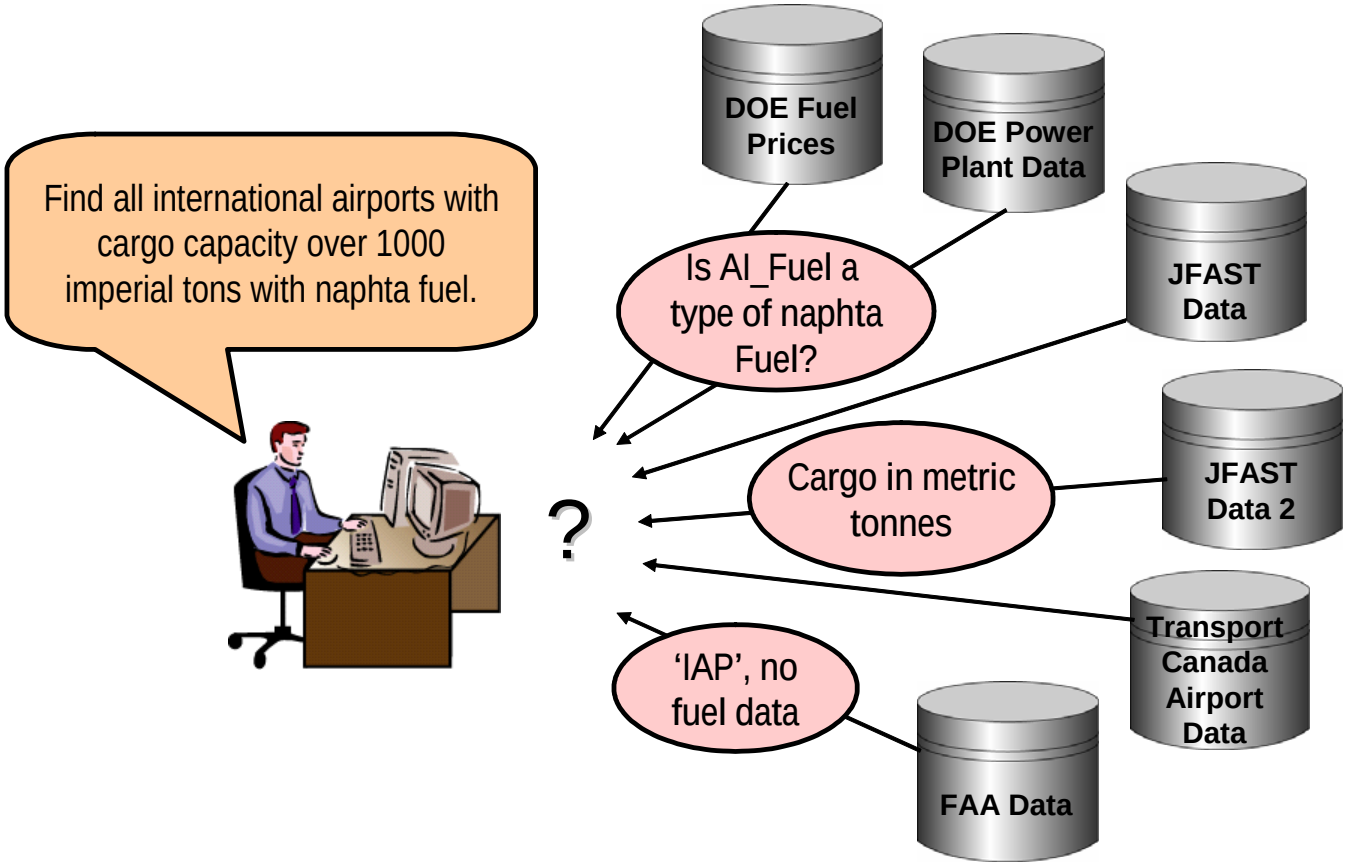
Semantic Interoperability

- "Semantic interoperability is defined as the enablement of software systems ... to interoperate at a level in which the exchange of information is at the enterprise level. This means each system (or object of a system) can map from its own conceptual model to the conceptual model of other systems, thereby ensuring that the meaning of their information is transmitted, accepted, understood, and used across the enterprise." –Obrst *et al*

- How and to what extent do ontologies facilitate *semantic interoperability*?

AIXE: IET's Semantic Integration Tool

- Information Extraction & Transport, Inc. (IET) is developing the Application Information Exchange Environment (AIXE), as a Phase II SBIR for the Navy, to:
  - allow users to quickly map new, dynamic and legacy data sources to the system.
  - integrate diverse data at query time to generate a single integrated data/knowledge base for answering queries.


Find all international airports with cargo capacity over 1000 imperial tons with naphta fuel.
?
DOE Fuel Prices
DOE Power Plant Data
JFAST Data
Is Al_Fuel a type of naphta Fuel?
JFAST Data 2
Cargo in metric tonnes
Transport Canada Airport Data
'IAP', no fuel data
FAA Data

AIXE General Approach

- Use an ontology/logic-based foundational data scheme that implements OWL markup plus other tools (translation scripts, Bayesian reasoning) for interoperability
- Simple ontology and logic-aided schema extension tool that logic-naïve users can implement

AIXE General Approach

- We extend the central ontology as necessary for each new data source (database tables, spreadsheets, structured web pages, etc.) and then define a translation scheme to wrap (or rewrap) the data sources with Class and property wrappers from the central ontology.
- For each data source, we define a mapping to our ontology on a field by field basis.

1	GEO	NAME	TYPE	CC	LAT	LON	RLAT	RLON	CINC	ICAO	COUNTRY	USEF
2	AFSD	Aalmeria	PRT	SP	365000N	0022800W	0.642863	4.31E-02	4		Spain	
3	AFSF	Aalmelo	STG	NL	522000N	0063800E	0.913389	-0.11577	4		Netherland	
4	AFSJ	Aalmena	CTY	20	395332N	0994224W	0.696251	1.74021	7		Kansas	
5	AFSL	Almeria	CTY	31	414933N	0993118W	0.729998	1.736981	7		Nebraska	
6	AFSM	Abdullex	MAP	ID	264240N	1205623W	0.832716	2.07589	7	WISI	Indonesia	
7	AFSN	Abbeville	MAP	FR	410601N	13758W	0.734788	1.983266	7	LFOW	France	
8	AFSP	Alimo	APT	23	364141N	0881642W	0.640444	1.540748	7	ETMA	Kentucky	
9	AFSQ	Alimond	CTY	9	353905N	0915850W	0.622234	1.605364	7		Arkansas	
10	AFSR	Alimond	CTY	16	421920N	0774410W	0.738662	1.356795	7		New York	
11	AFSS	Alimelo	CTY	NL	522157N	0064060E	0.913956	-0.1165	4		Netherland	
12	AFSV	Alimont	CTY	7	383953N	1065044W	0.674827	1.864807	7		Colorado	
13	AFSW	Alimont	CTY	22	415514N	0860242W	0.719105	1.239409	7		Michigan	
14	AFSX	Alimont	CTY	32	464331N	1013008W	0.81551	1.771548	7		North Dak	
15	AFSY	Almira	CTY	9	342422N	0912434W	0.6005	1.595396	7		Arkansas	
16	AFT1	Almiranto	PRT	PM	091700N	0822300W	0.162025	1.43786	6		Panama	

```
<rdf:Description rdf:about="#AFSP">
 <rdfs:label>Alimo</rdfs:label>
 <rdf:type rdf:resource="&aixeFds;#APT"/>
 <aixeFds:locationOfObject rdf:resource="&aixeFds;#ST23"/>
 <aixeFds:latitudeNumD rdf:datatype="&aixeFds;#LatLongDAFormat">41060N</aixeFds:latitudeNumD>
 <aixeFds:longitudeNumD rdf:datatype="&aixeFds;#LatLongDAFormat">0881642W</aixeFds:longitudeNumD>
 <aixeFds:latitudeRad rdf:datatype="&aixeFds;#LatLongRadians">0.640444</aixeFds:latitudeRad>
 <aixeFds:longitudeRad rdf:datatype="&aixeFds;#LatLongRadians">1.540748</aixeFds:longitudeRad>
 <aixeFds:icaoCode>ETMA</aixeFds:icaoCode>
</rdf:Description>
```

The mapping allows us to convert the data into AIXE format when we need it.

Reasoning Applications

- Identifying infrastructure objects in a given area
- Identifying potential dependencies
- Analyze “what if” scenarios.
- Collecting all information relevant to a particular object, location, etc.

- This presentation: Consider challenges that arise in integrating disparate data
  - How does the ontology and supporting inference tools ease integration of disparate data and what are the limitations?
  - Consider in terms of example questions that we might pose to the system

- Interoperability Issues
  - Identity and Glossary Control
  - Power of Transitivity Reasoning
  - The Space Carving Problem
  - Up and Down the Subclass Hierarchy (Granularity, Part 1)
  - Faceting
  - Combining Hierarchies
  - Format and Unit Translation
  - Granularity, Part 2
  - Credibility

Identity and Glossary Control

- Example Query: Find all civilian airports selling fuel of type F12

Suppose that other data sources use different labeling convention for fuel types, i.e., they refer to F-12 fuel with a different name. This points to an obvious ontology application, call it glossary control, the management of different labels for single objects and managing the polysemy of labeling terms.

CODE	FLIP	NATO	AKA	EEFC	DEFINITION
A	115	F-22		BA	115/145 octane gasoline, leaded, MIL-L-5572F (PURPLE)
B	100	None			100/130 octane gasoline, leaded, MIL-L-5572F (GREEN)
C	None	None		B91	91/96 octane gasoline, leaded, No MIL Spec.
D	80	F-12	887		80/87 octane gasoline, leaded, MIL-L-5572F (RED)
F	None	None	80NL		80 octane gasoline, unleaded, No MIL Spec.
G	None	None	AvGas		Aviation Gasoline (AVGAS), octane unknown.
H	None	None			108/135 octane gasoline, leaded, No MIL Spec.
K	None	None	73NL		73 octane gasoline, unleaded, No MIL Spec.
L	100LL F-18	B95,B100			100/130 MIL Spec, low lead, aviation gasoline (BLUE)

- Approaches to glossary control: (i) Reify a new object for each term used, and use identity reasoning **or** (ii) attach different labels to single objects?

```
:Flip_80                                    (i)
  a   owl:Class;
  rdfs:subClassof  LowOctaneGasoline.

:Nato_F12
  a   owl:Class;
  owl:equivalentClass    Flip_80.

:AKA_887
  a   owl:Class;
  owl:equivalentClass    Flip_80.
```

```
:Flip_80                                    (ii)
  a   owl:Class;
  rdfs:subClassof     LowOctaneGasoline.
  natoLabel:  “Nato_F12”;
  akaLabel:   “AKA_887”;
  flipLabel:  “Flip_80”.

:natoLabel
   a      owl:AnnotationProperty;
   rdfs:subPropertyOf     rdfs:label.
```

- Approach (i) to glossary control:
  - Use annotation properties:
    - Simply map each term to the object via ‘label’ or create subproperties of ‘label’ that allow us to quickly distinguish different labeling sources.
      - e.g., (subProperty natolabel label).
      - This keeps our ontology lean and mean, distinguishing annotation issues from reasoning and representation issues.

- Challenges:
  - This is a straightforward way to realize the interoperability but it becomes more difficult to use the data implementing that label or query using the terms.
    - Consider, if our data source indicates that

      (fuelTypeAvailable Airport639 AKA_887)

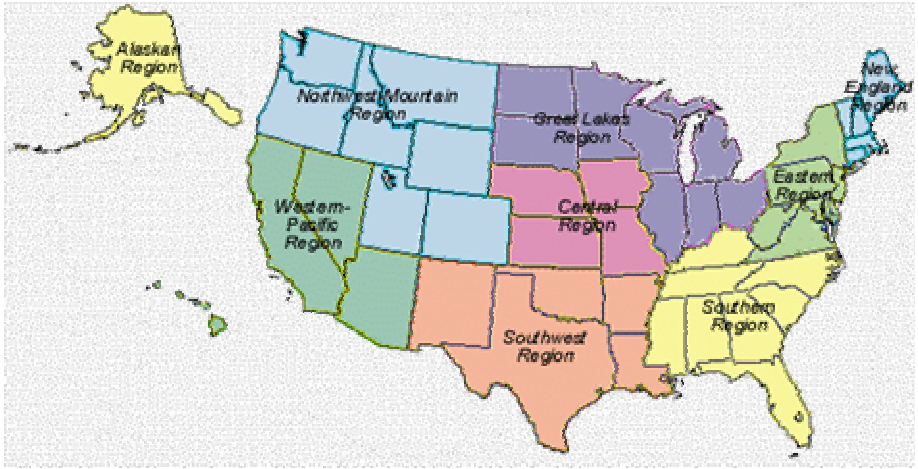
    if “AKA_887” is just a label in our ontology, we need to replace it with a direct reference to the object that it denotes, i.e., Flip_80. Similarly, “AKA_887” can’t be directly used in queries if it’s only a label, not a direct denotation of a reified object.

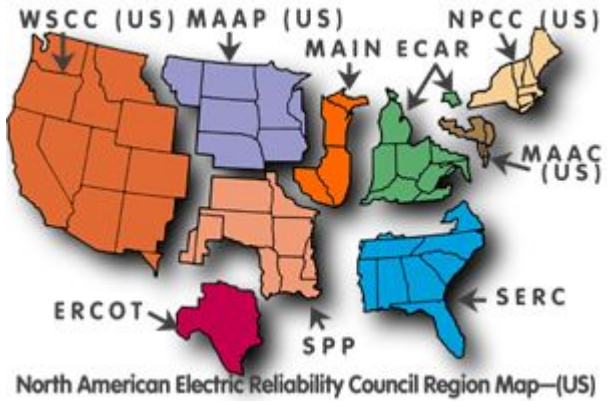
- Approach (ii) to glossary control:
  - Reify an object for each new name and then declare them as identical.
    - This simplifies data transformation and querying.
  - Challenges:
    - This may complicate inferencing depending on means of supporting identity reasoning, by dramatically increasing the size of the knowledge base or failing to support all the identity reasoning.
    - We conflate annotation issues with representation issues in our ontology.

Power of Transitivity and “Space Carving”

- Example Query: Find any objects in Western Pacific FAA region dependent on objects in NERC Region, SPP.

- Two Challenges:
  - Dependency linkages
  - Integrate the asset location and dependency information with information about two distinct federal region breakdowns, i.e., FAA regions and NERC regions. There are many ways to subdivide the physical regions into subregions and our system must reason across each.





Power of Transitive Reasoning

	A	B	C	D	E	F	G
1	DEPENDENCYID	ASSETID	SUPPORTSDEPENDENCYID	STARTTIM	ENDTIME	CRITICAL	RISK
2	1077	5161	1072				
3	1078	5723	1077				
4	1079	5612	978				
5	1080	5340	1079				
6	1081	5320	1080				
7	1082	5300	1079				
8	1084	5056	1083				
9	1085	5660	983				
10	1072	5910	1085				
11	1087	5911	1086				
12	1088	5891	1085				
13	1089	5561	1085				
14	1090	5560					
15	983	5066					
16	1092	5910					
17	1093	5191	1092				
18	1094	5189	1091				

→ *(dependentOn 5723 5066)*

Suppose this table specifies dependencies between assets.
We can extend the reasoning by enforcing the transitivity of dependence.
This query is more difficult in straight SQL, easy with transitive reasoning.

Space Carving

- The integration challenge arises from the need to integrate asset information with different geographical information.
  - (location ASSET_5066 City345)
  - (subRegionOf City345 New Mexico)
  - (subRegionOf New_Mexico SPP) → (location Asset_5066 SPP)

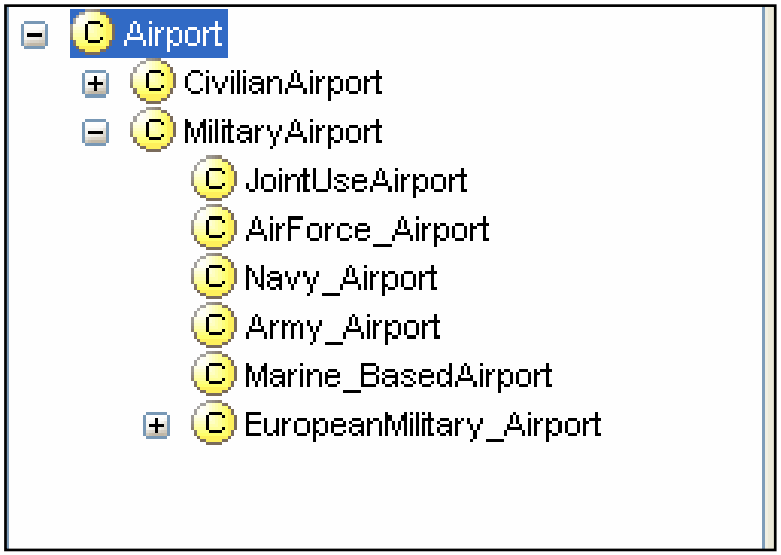
  - (location ASSET_5723 City234)
  - (subRegionOf City234 California)
  - (subRegionOf California Western_Pac_Reg) → (location Asset_5723 WPR)
- The integration of different “space carvings” requires:
  - That the ontology contain the high level parts in terms of which we can define the distinct space carvings.
  - The ability to represent and reason about the transitive parthood relations, i.e., that B’s parts are A’s parts if B is part of A.

Up and Down the Subclass Hierarchy

- Example Query: Find all military airports in the northwest

- Challenge: The challenge here lies in the fact that some data sources distinguish between air force airports, naval airports and other DOD-controlled airports. Similarly, some distinguish between joint-use airports (military and civilian) and military airports. Others simply distinguish between military and civilian airports. (Also, system needs to integrate geographical information and recognize all parts of the northwest.)

- This is addressed rather straightforwardly, i.e., by utilizing subtyping.



Interested users can query at the desired level of specificity. However, a more general query will also capture instances of more specific subclasses. The utilization of hierarchies overcomes *some* of the challenges associated with representations at different granularity levels.

Note that the class hierarchy also allows users to quickly extend the ontology and map to existing schemes. And, users can do extensive querying with a lot of ignorance of the original data schemes.

Faceting

- Example Query: Find [city, airport, fuel type] most similar to [city, airport, fuel type] X.

- Challenge:
  - Different data schemes carve up concepts with respect to different properties. Airport subtyping might be done with respect to location, size, functionality, etc. Similarly, fuel typing might be done in terms of basic chemical makeup (e.g., kerosene vs. gasoline) and/or kinds and levels of additives, (octane, lead, deicer).
  - Answering the above question, and integrating new data into the ontology depends on the ability to quickly determine the different ways in which the reasoning space is carved up.

- Integrating these different representational schemes requires hierarchical reasoning but also some kind of “faceting” or partitioning of the reasoning space. Ideally, our integration ontology allows us to partition or carve up the workspace in different ways. One solution, second order classes:
  - AirportsByFunction = {CivilAirport, MilitaryAirport, JointUseAirport, …}
  - AirportsBySize = {MetropolitanAirport, MidSizeAirport, SmallRegionalAirport, …}

- We must recognize what the different representations have in common, e.g., all are subclasses of airport, but also allows us to focus on different ways to subdivide the reasoning space.
- This approach requires both multiple inheritance and second order classes (beyond DL reasoners).
- This facilitates data retrieval and the mapping of new concepts into the domain, i.e., it becomes easier to find the different ways in which the domain is partitioned/faceted.

Combining Hierarchies

- Example Query: Find all training facilities in VA controlled by the DoD.

- Challenge: Relevant information is stored in up to four different data sources, i.e., geographical information about VA, subclass hierarchies about military infrastructure, parthood information about military infrastructure, and military organization charts.
- Here we're doing more than simple "isa" reasoning, we're trying to reason about the extent to which properties of the whole apply to the part, and vice versa.
  - Consider the DoD, many properties of its parts don't apply to the whole, but some do. We need to write more subtle rules to reason about this.
    - "All things controlled by suborganizations of an organization are controlled by the organization"
- This starts to push us further beyond simple DL-based ontologies, this is most easily accomplished with horn rules or other representation and reasoning tools beyond DLs.

Ontology Limitations

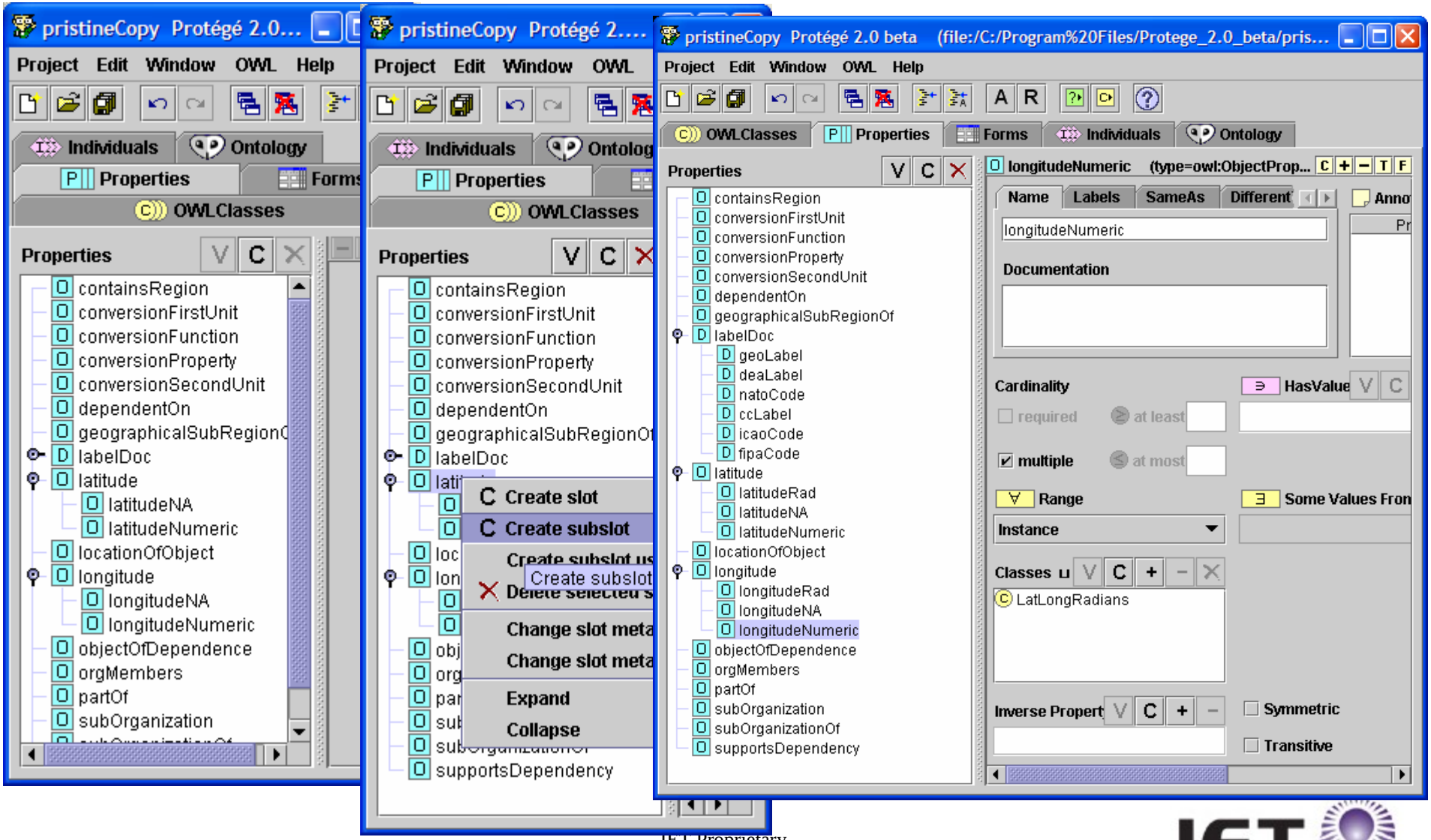
- Obviously, the more reasoning we can do the easier it is to query and integrate disparate data sources, but what kinds of things can't we do with ontologies alone?
  - Different formats
  - Some granularity challenges
  - Credibility reasoning

Format and Unit Translation

- Example Query: Find all military assets between 70 and 55 W and 30 and 40 N capable of carrying over 100 metric tonnes/day.

- Challenge: One of our data sources represent location information in terms of radians, and most of them represent cargo capacity in terms of imperial tons.
- How can an ontology help here?
  - We use the ontology to track datatypes and create datatype property hierarchies for purposes of guiding calls to translation tools.

Subproperty hierarchies are used to guide translation



Format and Unit Translation

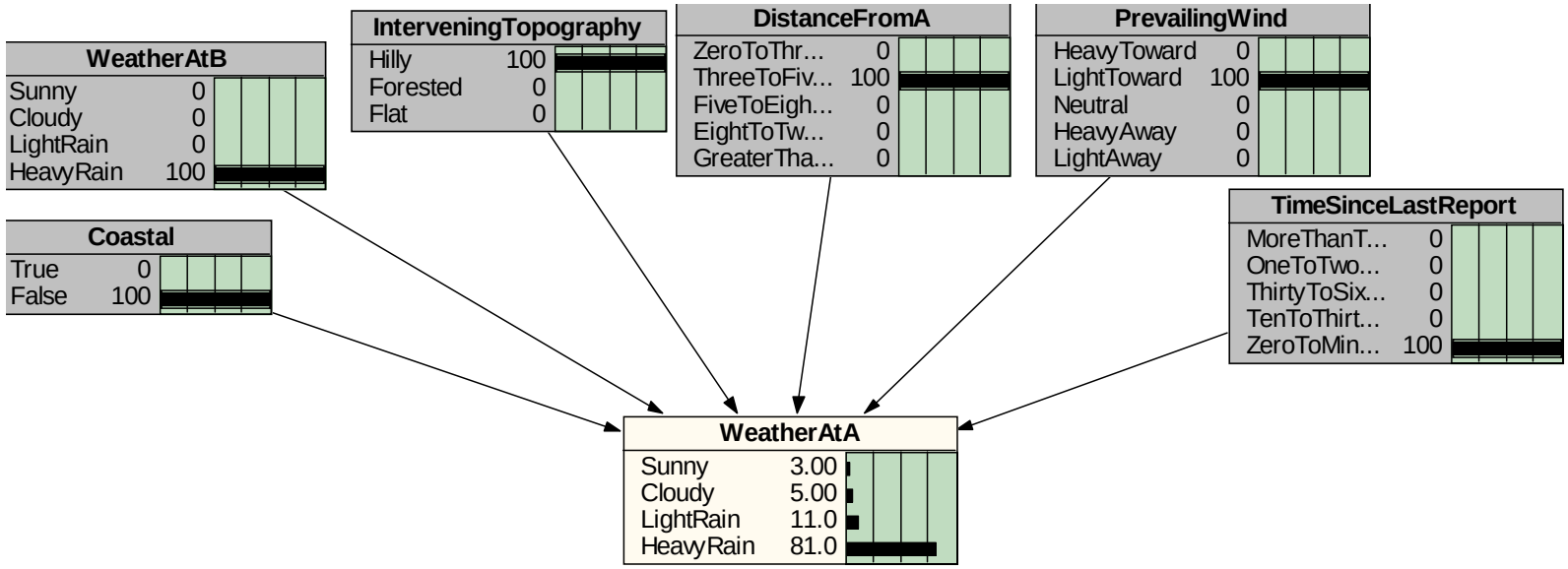
- Parse query and remove all translatable properties
  - Subquery to determine relevant “sibling” properties.
  - (<aixeFds:latitudeDegree>,<rdf:subPropertyOf> ?X)(?PROP rdfs:subPropertyOf ?PROP)
  - And look for property pairs for which a translation function is defined
- SELECT ?apt ?lat ?long WHERE
  - (?apt, <rdf:type>,<aixeFds:Military-Airport>)(<aixeFds:latitude> ?apt ?lat)(aixeFds:longitude,?apt,?long)
  - Note that this will return latitudeNumeric, latitudeRad and latitudeNA (these are the subproperties) and then we invoke appropriate translation tools. Ontology helps to render the search reasonable.

Granularity, Part 2

- Example Query: Describe terrain at region3352
- Example Query: Is it raining at location T?

- Challenge:
  - We know the terrain in three subregions of X, how do we integrate that into a terrain assessment for X?
  - We know weather in three different locations surrounding T, how do we approximate weather at T?

- Here we may have to resort to other reasoning means to reason from one granularity level to another or to reapply known information to the question at hand:



Credibility

- Consider other challenges:
  - How do we resolve contradictory or differing reports from amongst the different data sources?

- Challenge:
  - Use metadata to evaluate new data sources